

## Contents

<b>1 Routine/Function Prologues</b>	<b>2</b>
1.0.1 vic_run.c (Source File: vic_run.c) . . . . .	2
1.0.2 calc_moist_by_depth (Source File: vic_run.c) . . . . .	5
1.0.3 calc_max_moist_by_depth (Source File: vic_run.c) . . . . .	6

## 1 Routine/Function Prologues

### 1.0.1 vic\_run.c (Source File: vic\_run.c)

Executes VIC model runs on each tile

INTERFACE:

```
void FTN(vic_run)(int *nlayer,
                  int *nnode,
                  int *snowband,
                  int *day,
                  int *day_in_year,
                  int *hour,
                  int *month,
                  int *year,
                  int *full_energy_flag,
                  int *frozen_soil_flag,
                  int *grnd_flux_flag,
                  int *quick_flux_flag,
                  int *outputflag)
{
```

CONTENTS:

```
dmy.day = *day;
dmy.day_in_year = *day_in_year;
dmy.hour = *hour;
dmy.month = *month;
dmy.year = *year;
for ( i = 0; i < tspmd.cdi_array[par.rank]; i++)
{
    full_energy(i,&atmos[i], &soil_con[i], veg_con[i], &prcp[i],
dmy, *nlayer, *nnode, *snowband, *full_energy_flag,
*frozen_soil_flag,
*grnd_flux_flag, *quick_flux_flag);
}
#endif
for(i=0; i< tspmd.cdi_array[par.rank]; i++){
    outdata1[i].count += 1;
    energy = prcp[i].energy;
    cell = prcp[i].cell;
    veg_var = prcp[i].veg_var;
    snow = prcp[i].snow;
    for(veg=0; veg<= 1; veg++){
        outdata1[i].acond += cell[WET][veg][0].aero_resist[0];
        for(band=0; band<*snowband; band++){
outdata1[i].swnet += energy[veg][band].shortwave;
outdata1[i].lwnet += energy[veg][band].longwave;
outdata1[i].qle -=energy[veg][band].latent;
```

```

outdata1[i].qh +=energy[veg][band].sensible;
outdata1[i].qg +=energy[veg][band].grnd_flux+
    energy[veg][band].deltaH;
outdata1[i].qfz +=energy[veg][band].refreeze_energy;
}
tmp_rainf =calc_rainonly(atmos[i].air_temp,atmos[i].prec,
1.0);
    outdata1[i].rainf += tmp_rainf;
    outdata1[i].snowf +=(atmos[i].prec-tmp_rainf);
}
for(veg=0; veg<= 1; veg++){
    for(dist=0; dist<Ndistr; dist++){
for(band=0; band<*snowband; band++){
    tmp_depth = 0.0;
    for(index=0; index< *nlayer; index++){
        tmp_depth +=soil_con[i].depth[index];
    }
    tmp_moist_sum = calc_moist_by_depth(tmp_depth,
        cell[dist][veg][band].layer,soil_con[i].depth, *nlayer);
    tmp_max_moist_sum = calc_max_moist_by_depth(tmp_depth,
        soil_con[i].porosity,soil_con[i].depth, *nlayer);
    outdata1[i].soilwet +=(tmp_max_moist_sum==0.0?0.0:
    tmp_moist_sum/tmp_max_moist_sum);
    tmp_depth = 0.0;
    if(veg < 1){
        for(index = 0; index < 2; index++)
            tmp_depth +=veg_con[i][veg].zone_depth[index];
        tmp_moist_sum = calc_moist_by_depth(tmp_depth,
            cell[dist][veg][band].layer, soil_con[i].depth, *nlayer);
        tmp_max_moist_sum = calc_max_moist_by_depth(tmp_depth,
            soil_con[i].porosity, soil_con[i].depth, *nlayer);
    }
    else
        tmp_moist_sum = 0.0;
    outdata1[i].rootmoist +=tmp_moist_sum;
}
}
    outdata1[i].soilwet /=2;
}
for(veg=0; veg< 1; veg++){
    tmp_evap = 0.0;
    surf_temp = 0.0;
    rad_temp = 0.0;
    albedo = 0.0;
    t_veg = 0.0;
    tmp_moist = 0.0;
    tmp_ice = 0.0;
    swe = 0.0;
}

```

```

        for(band=0; band<*snowband; band++){
tmp_evap+= snow[veg][band].vapor_flux*1000+
    snow[veg][band].canopy_vapor_flux*1000;
surf_temp +=energy[veg][band].T[0]+273.16;
if(snow[veg][band].swq > 0)
    rad_temp+=(snow[veg][band].surf_temp+273.16);
else
    rad_temp+=(energy[veg][band].T[0]+273.16);
albedo+=energy[veg][band].albedo;
swe+=snow[veg][band].swq;
outdata1[i].qsm +=snow[veg][band].melt/(dmy.dt);
for(dist=0; dist<Ndist; dist++){
    for(index=0; index < *nlayer; index++){
        t_veg +=cell[dist][veg][band].layer[index].evap;
        if(veg < 1)
            outdata1[i].tveg = t_veg;
        else
            outdata1[i].esoil = t_veg;

        tmp_evap+=cell[dist][veg][band].layer[index].evap;
        tmp_moist = cell[dist][veg][band].layer[index].moist;
        tmp_ice = cell[dist][veg][band].layer[index].ice;
        tmp_moist -=tmp_ice;
        outdata1[i].moist[index] =tmp_moist;
        outdata1[i].qs+=cell[dist][veg][band].runoff/(dmy.dt);
        outdata1[i].qsb+=cell[dist][veg][band].baseflow/(dmy.dt);
    }
    tmp_evap+=veg_var[dist][veg][band].canopyevap;
}
outdata1[i].snowt +=snow[veg][band].surf_temp;
}
outdata1[i].avgsurft = surf_temp;
outdata1[i].radt = rad_temp;
outdata1[i].evap +=tmp_evap;
outdata1[i].albedo = albedo;
outdata1[i].swe = swe;
if(*outputflag){
outdata1[i].moist_prev = outdata1[i].moist[0] +
    outdata1[i].moist[1]+outdata1[i].moist[2];
outdata1[i].swe_prev = outdata1[i].swe;
}
}
}
#endif

```

---

### 1.0.2 calc\_moist\_by\_depth (Source File: vic\_run.c)

This function calculates the soil moisture depth from the surface to a given depth. It assumes that the soil moisture is evenly distributed over a soil layer (which it is, I think).

INTERFACE:

```
float calc_moist_by_depth(float the_depth,
                           layer_data_struct *layer,
                           float *depth, int Nlayer)
```

CONTENTS:

```
moist = 0.0;
depth_total = 0.0;
//-----
// loop through the soil layers from top to bottom
//-----
for(index=0;index<Nlayer;index++) {
//-----
// calculate the depth to the bottom of the current layer
//-----
depth_total += depth[index];
//-----
// compare with the depth in question
//-----
if(the_depth > depth_total) {
//-
// the depth is past the current layer
//-
moist += layer[index].moist;
}

else {
//-
// the depth is within the current layer
//-
moist += layer[index].moist * (the_depth -
(depth_total - depth[index]))/depth[index];
break;
}

}

return(moist);
```

---

### 1.0.3 calc\_max\_moist\_by\_depth (Source File: vic\_run.c)

This function calculates the max soil moisture depth from the surface to a given depth. It assumes that the soil moisture is evenly distributed over a soil layer (which it is, I think). Max allowable soil moisture = depth \* porosity.

INTERFACE:

```
float calc_max_moist_by_depth(float the_depth,
                               float *porosity,
                               float *depth, int Nlayer)
```

CONTENTS:

```
moist = 0.0;
depth_total = 0.0;
//-----
// loop through the soil layers from top to bottom
//-----
for(index=0;index<Nlayer;index++) {
    //-----
    // calculate the depth to the bottom of the current layer
    //-----
    depth_total += depth[index];
    //-----
    // compare with the depth in question
    //-----
    if(the_depth > depth_total) {
        moist += porosity[index] * 1000. * depth[index];
    }
    else {
        moist += porosity[index] * 1000. *
depth[index] * (the_depth - (depth_total
- depth[index]))/depth[index];
        break;
    }
}
return(moist);
```